The data structure

```
1  class DJSets { vector<int> s; };
2
3  DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5  void DJSets::unionSets( int rA, int rB ) {
6      if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
7      else {
8          if( s[ rA ] == s[ rB ] ) --s[ rA ];
9          s[ rB ] = rA;
10     }
11 }
12
13 int DJSets::find( int x ) {
14     if( s[ x ] < 0 ) return x;
15     else    return find( s[ x ] );
16 }
```

The visualization

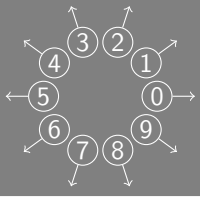
main

```
// BEGIN
DJSets djSet{10};
cout << djSet.find(4);
```

vector s

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else    return find( s[ x ] );
17  }
```

The visualization

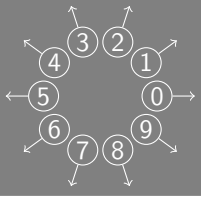
main

```
DJSets djSet{10};
cout << djSet.find(4); // 4
cout << djSet.find(2);
```

vector s

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The graph




```
1  class DJSets { vector<int> s; };
2
3  DJSets::DJSets( int n ) : s( n, −1 ) { }
4
5  void DJSets::unionSets( int rA, int rB ) {
6      // assume: rA, rB roots and different
7      if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8      else {
9          if( s[ rA ] == s[ rB ] ) −−s[ rA ];
10         s[ rB ] = rA;
11     }
12 }
13
14 int DJSets::find( int x ) {
15     if( s[ x ] < 0 ) return x;
16     else     return find( s[ x ] );
17 }
```

The visualization

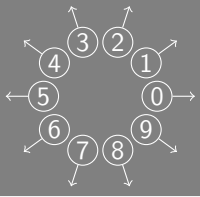
main

```
cout << djSet.find(4); // 4
cout << djSet.find(2); // 2
djSet.unionSets(4, 2);
```

vector s

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else     return find( s[ x ] );
17  }
```

The visualization

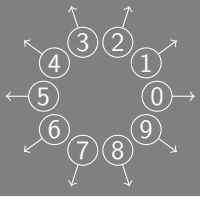
main

```
cout << djSet.find(2); // 2
djSet.unionSets(4, 2);
cout << djSet.find(4);
```

vector s

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else    return find( s[ x ] );
17  }
```

The visualization

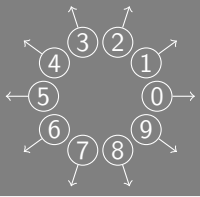
main

```
cout << djSet.find(2); // 2
djSet.unionSets(4, 2);
cout << djSet.find(4);
```

vector s

-1	-1	-1	-1	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else    return find( s[ x ] );
17  }
```

The visualization

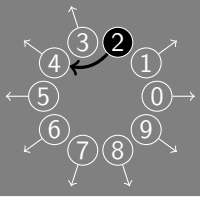
main

```
cout << djSet.find(2); // 2
djSet.unionSets(4, 2);
cout << djSet.find(4);
```

vector s

-1	-1	4	-1	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else    return find( s[ x ] );
17  }
```

The visualization

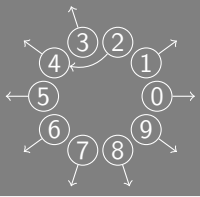
main

```
djSet.unionSets(4, 2);
cout << djSet.find(4); // 4
cout << djSet.find(2);
```

vector s

-1	-1	4	-1	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else     return find( s[ x ] );
17  }
```

The visualization

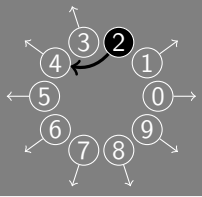
main

```
cout << djSet.find(4); // 4
cout << djSet.find(2); // f(4)
djSet.unionSets(4, 3);
```

vector s

-1	-1	4	-1	-2	-1	-1	-1	-1	-1

The graph




```
1  class DJSets { vector<int> s; };
2
3  DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5  void DJSets::unionSets( int rA, int rB ) {
6      // assume: rA, rB roots and different
7      if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8      else {
9          if( s[ rA ] == s[ rB ] ) --s[ rA ];
10         s[ rB ] = rA;
11     }
12 }
13
14 int DJSets::find( int x ) {
15     if( s[ x ] < 0 ) return x;
16     else     return find( s[ x ] );
17 }
```

The visualization

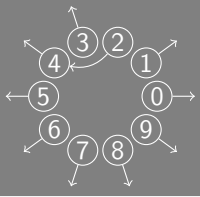
main

```
cout << djSet.find(4); // 4
cout << djSet.find(2); // 4
djSet.unionSets(4, 3);
```

vector s

-1	-1	4	-1	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, −1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) −−s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else     return find( s[ x ] );
17  }
```

The visualization

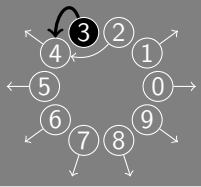
main

```
cout << djSet.find(2); // 4
djSet.unionSets(4, 3);
cout << djSet.find(3); // f(4)
```

vector s

-1	-1	4	4	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, -1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) --s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else     return find( s[ x ] );
17  }
```

The visualization

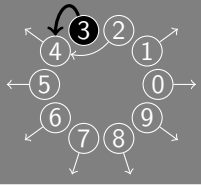
main

```
djSet.unionSets(4, 3);
cout << djSet.find(3); // f(4)
// END
```

vector s

-1	-1	4	4	-2	-1	-1	-1	-1	-1

The graph




```
1   class DJSets { vector<int> s; };
2
3   DJSets::DJSets( int n ) : s( n, −1 ) { }
4
5   void DJSets::unionSets( int rA, int rB ) {
6       // assume: rA, rB roots and different
7       if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8       else {
9           if( s[ rA ] == s[ rB ] ) −−s[ rA ];
10          s[ rB ] = rA;
11      }
12  }
13
14  int DJSets::find( int x ) {
15      if( s[ x ] < 0 ) return x;
16      else    return find( s[ x ] );
17  }
```

The visualization

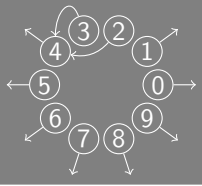
main

```
djSet.unionSets(4, 3);
cout << djSet.find(3); // 4
// END
```

vector s

-1	-1	4	4	-2	-1	-1	-1	-1	-1

The graph




```
1  class DJSets { vector<int> s; };
2
3  DJSets::DJSets( int n ) : s( n, −1 ) { }
4
5  void DJSets::unionSets( int rA, int rB ) {
6      // assume: rA, rB roots and different
7      if( s[ rB ] < s[ rA ] ) s[ rA ] = rB;
8      else {
9          if( s[ rA ] == s[ rB ] ) −−s[ rA ];
10         s[ rB ] = rA;
11     }
12 }
13
14 int DJSets::find( int x ) {
15     if( s[ x ] < 0 ) return x;
16     else     return find( s[ x ] );
17 }
```